THE GENERALIZED NODE-CAPACITATED MAXIMUM FLOW PROBLEM

Cenk Çalışkan, Woodbury School of Business, Utah Valley University, 800 W. University Pkwy, Orem, UT 84058, (801) 863-6487, cenk.caliskan@uvu.edu

ABSTRACT

The generalized node-capacitated maximum flow problem is to send the maximum amount of flow from a source node to a sink node in a directed network with node capacities, where flow on a given arc exerts a workload proportional to its amount on its tail and head nodes; and sum of all such workloads on each node cannot exceed the node's capacity. The problem has important applications and efficient solution methods are essential. We develop an efficient Lagrangian relaxation and branch and bound based method that solves the problem as a series of shortest path problems.

Keywords: maximum flow problem, lagrangian relaxation, networks, branch-and-bound, subgradient optimization

INTRODUCTION

The classical maximum flow problem is well-known and extensively studied in operations research. The first solution algorithm for the problem was proposed by Ford and Fulkerson [8]. It has many applications, either directly or as a subproblem within the context of larger problems in transportation, logistics, telecommunications, and many other fields. To this date, researchers have proposed many efficient algorithms for the classical maximum flow problem. Ahuja et al. [2] describe many algorithms and numerous applications for the classical maximum flow problem. Fulkerson [9] describes a maximum flow problem with an additional convex budget constraint to represent a network expansion problem. Malek-Zavarei and Frisch [14] considers a version of the maximum flow problem which is related to the problem we study in this paper. In that problem, there are linear constraints on some of the nodes: linear combination of flows entering some of the nodes are limited by node capacities. They propose a Lagrangian relaxation based heuristic approach. Ahuja and Orlin [1] study another type of maximum flow problem in which there is a budget constraint limiting the arc flows and propose a capacity scaling algorithm. Çalışkan [4] shows that the algorithm of Ahuja and Orlin [1] may sometimes terminate with an infeasible flow and proposes modifications in the algorithm. Çalışkan [3, 5, 6] propose double scaling, cost scaling and specialized network simplex algorithms for the problem, respectively.

We study another variant of the maximum flow problem in which flow on a given arc exerts a workload proportional to its amount on its tail and head nodes; and sum of all such workloads on each node must be less than or equal to the node's capacity. This is a generalization of the problem studied by Malek-Zavarei and Frisch [14], for which they describe a number of applications in computer networks. Our motivation came from a problem studied by Hall and Lotspeich [12] and Hall and Çalışkan [11] in which a highway is modeled as a lattice of nodes and directed arcs, and

the objective is to maximize the flow through the highway subject to node capacity constraints where total workload exerted on each node by flows passing through the node is limited by node capacity.

FORMULATION OF THE PROBLEM

Let $G = (N, A)$ be a directed network where N is the set of nodes and A is the set of arcs. In this network, the flow on each arc (i, j) is represented by the nonnegative variable x_{ij}; and it exerts workload α_{ij} on tail node i and γ_{ij} on head node j, per unit of flow. We call α_{ij} the "*out-workload*" and γ_{ij}, the "*in-workload*" coefficient of arc (i, j). For simplicity of exposition, we add an artificial arc (t, s) with $\alpha_{ts} = \gamma_{ts} = 0$. The generalized node-constrained maximum flow problem is to send the maximum amount of flow from a source node s to a sink node t where the sum of flows into and out of each node $i \in N$ are equal and the total workload for each node $i \in N$ is limited by W_i, the capacity of node i. The problem then can be formulated as follows.

[P1] $\min z = -x_{ts}$ (1)

$s.t.$

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{j,i} = 0 \qquad \forall i \in N \qquad (2)$$

$$\sum_{(i,j)\in A} \alpha_{ij} x_{ij} + \sum_{(j,i)\in A} \gamma_{ji} x_{ji} \leq W_i \qquad \forall i \in N \qquad (3)$$

$$x_{ij} \geq 0 \quad and \quad integer \qquad \forall (i,j) \in A \qquad (4)$$

For reasons that will be clear later on in the paper, we chose to express the problem as minimization as opposed to maximization. Equation 3 replaces the arc capacity constraints of the classical maximum flow problem. Equations 2 and 4 are pure network constraints. Equation 3 is the set of complicating constraints or side constraints, which cause the loss of nice combinatorial properties of the maximum flow problem. Network flow problems with side constraints may be solved very efficiently in practice using variants of the simplex algorithm that exploit the network substructure present in the coefficient matrix of the LP. Klingman and Russell [13], Graves and McBride [10], Chen and Saigal [7], and Mamer and McBride [15] present specialized simplex algorithms for the problem of networks with side constraints. Efficiency of these approaches primarily depend on the fact that the number of side constraints is usually much less than the number of nodes of the network in practice. These methods carry a "working basis" which is a $k \times k$ matrix, where k is the number of side constraints. In the generalized node-capacitated maximum flow problem, k is equal to the number of nodes, so the size of the working basis is equal to that of the network basis. We don't expect too much speed-up in the simplex algorithm from these methods that exploit the network substructure in this case.

A SPECIAL CASE

Consider a network (N, A) where $\alpha_{i,j} = \alpha_i$ and $\gamma_{j,i} = \gamma_i \qquad \forall i \in N$. In this case, flows going through each node i will exert the same total workload on it, regardless of where they came from and where they are going to; the workload per unit of flow on i will be $\alpha_i + \gamma_i$. Thus, the total flow that can pass through i is bounded by $W_i/(\alpha_i + \gamma_i)$. We can add these flow upper bounds on nodes and drop Equation 3. The resulting problem can be transformed into the classical maximum flow problem by replacing each node i by two nodes, i' and i'', putting an arc between

i' and i'' with capacity equal to the node flow-through capacity (it needs to be rounded down to ensure integral flows), and adding all incoming arcs of i to i' as incoming arcs, and adding all outgoing arcs of i to i'' as outgoing arcs. Figure 1 illustrates this conversion approach.

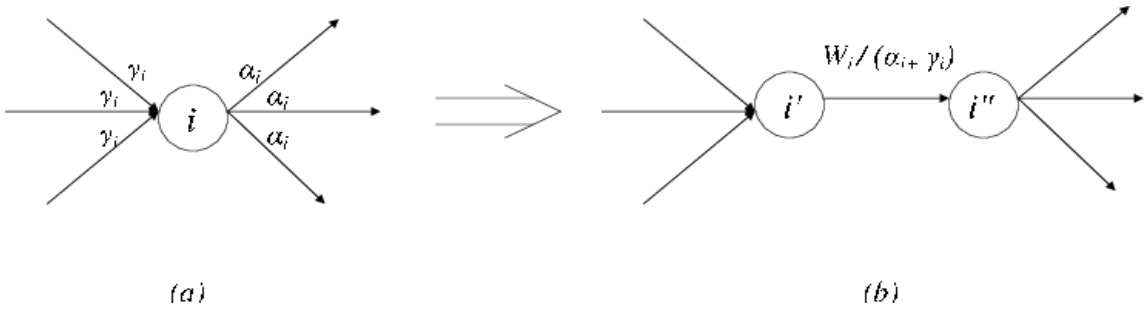

Figure 1: Conversion to classical maximum flow problem

LAGRANGIAN RELAXATION

If we relax Equation 3, the resulting problem is a minimum cost network flow problem. We define the Lagrange multipliers λ_i for $i \in N$ corresponding to Equation 3. Let X be the set of x that satisfy Equations 3-4. We then relax Equation 3 and bring it into the objective function, resulting in the following Lagrangian subproblem or Lagrangian function:

$$L(\lambda) = \min_{\lambda \geq 0}\{-x_{ts} + \sum_{i \in N} \lambda_i(\sum_{(i,j)\in A} \alpha_{ij}x_{ij} + \sum_{(j,i)\in A} \gamma_{ji}x_{ji} - W_i): x \in X\}.$$

This can be re-written as:

$$[P2] \qquad L(\lambda) = \min_{\lambda \geq 0}\{-x_{ts} + \sum_{(i,j)\in A} (\lambda_i\alpha_{ij} + \lambda_j\gamma_{ij})x_{ij} - \sum_{i \in N} \lambda_i W_i: x \in X\} \quad (5)$$

Let $\delta_i = \sum_{(i,j)\in A} \alpha_{ij}x_{ij} + \sum_{(j,i)\in A} \gamma_{ji}x_{ji} - W_i$ for $i \in N$. Then, Equation 3 can be re-written in the form $\delta \leq 0$.

Lemma 1. *For any $\lambda \geq 0$, the value $L(\lambda)$ of the Lagrangian function is a lower bound on the optimal objective function value z^* of problem P1.*

Proof: Let x^* be an optimal solution to problem P1 and $\hat{x}$ be an optimal solution to problem P2. Then,

$$z^* = -x_{ts}^* \geq -x_{ts}^* + \lambda(\alpha x^* + \gamma x^* - W) \geq -\hat{x}_{ts} + \lambda(\alpha\hat{x} + \gamma\hat{x} - W) \quad (6)$$

The first part of this inequality is because x^* is a feasible solution to problem P1, and the second part is because $\hat{x}$ is an optimal solution to problem P2. □

Solving the Minimum Cost Network Flow Problem

Node capacity constraints implicitly impose limits on arc flows. We can calculate upper bounds for all of the arc flows based on these implicit limits. Consider arc $(i,j) \in A$. The flow leaving

node i through arc (i,j) will impose a workload of at least $\omega_1 = \alpha_{ij} + min\{\gamma_{ki}|(k,i) \in A\}$. Similarly, the flow arriving at node j through arc (i,j) will impose a workload of at least $\omega_2 = \gamma_{ij} + min\{\alpha_{jk}|(j,k) \in A\}$. Thus, flow x_{ij} is bounded by $u_{ij} = \lfloor min\{W_i/\omega_1, W_j/\omega_2\} \rfloor$. We denote u_{ij} the *capacity* of arc $(i,j) \in A$, and $c_{ij} = \lambda_i \alpha_{ij} + \lambda_j \gamma_{ij}$ the cost of arc $(i,j) \in A$ for a given λ.

Definition 1. *Given a flow $x \in X$, we define the residual network $G(x)$ as follows. $G(x)$ contains the same node set as G. For each arc $(i,j) \in A$ with cost c_{ij}, capacity u_{ij} and flow x_{ij}:*

(i) if $x_{ij} = 0$, G(x) contains arc (i,j) with cost c_{ij} and capacity u_{ij}
(ii) if $x_{ij} = u_{ij}$, G(x) contains arc (j,i) with cost $-c_{ij}$ and capacity x_{ij}
(iii) if $u_{ij} > x_{ij} > 0$, G(x) contains arc (i,j) with cost c_{ij} and capacity $u_{ij} - x_{ij}$ and arc (j,i) with cost $-c_{ij}$ and capacity x_{ij}.

Definition 2. *Given a flow x and residual network $G(x)$, we denote a directed path from s to t in $G(x)$ an augmenting path.*

Theorem 1 (Malek-Zavarei and Frisch[14]). *The ε-augmentation of a minimum cost flow of value v from s to t along a minimum cost $s-t$ augmenting path yields a minimum cost of value $v + \varepsilon$.*

Theorem 1 can be used to solve the minimum cost flow problem for a given λ as follows. We start with $x = 0$ and find a minimum cost augmenting path. This can easily be accomplished by finding a shortest path from s to t in $G(x)$. We then augment the flow along this path as much as possible and repeat the procedure until no such path exists in $G(x)$. Note that the flow is optimal with respect to the given λ at each stage of this process.

Solving the Lagrangian Dual

Solving the minimum cost network flow problem as described above yields a lower bound for problem P1. In order to obtain the tightest lower bound, we maximize the Lagrangian function, i.e. we solve the following maximization problem, which is called the Lagrangian dual:

$$L^* = \max_{\lambda \geq 0} L(\lambda). \tag{7}$$

In order to solve the Lagrangian dual, we use a subgradient optimization procedure. We use an adaptation of the Newton's method in which we update the Lagrange multipliers in a relatively large step towards the optimal solution along the subgradient direction. The initial value of the multipliers are set to zero, and at every iteration, they are updated as follows:

$$\lambda_i^{t+1} = \max\{0, \lambda_i^t + \theta_t \delta_i\} \qquad \forall i \in N \tag{8}$$

where θ_t is the step size at iteration t. In order to make sure that the procedure will converge, the step size is chosen as follows (see, for instance, Ahuja et al. [2]):

$$\theta_t = \frac{\sigma_t[UB - L(\lambda^t)]}{||\delta||^2} \quad (9)$$

where $||\delta|| = (\sum_i \delta_i^2)^{0.5}$ is the Euclidean norm of δ and σ_k is a scalar that is chosen strictly between 0 and 2. The initial value of σ is 2, and it is halved once the Lagrangian objective function fails to increase after a long series of iterations. UB is the surrogate for the optimal Lagrangian objective function value, L^* and it is chosen as the best upper bound, i.e. the smallest objective function value of the feasible solutions found thus far in the algorithm.

procedure *solve_lagrangian_dual* $(G(x))$;
begin
  $z := -x_{ts} + \sum_{(i,j)\in A} c_{ij}$;
  $\lambda := 0$;
  $\sigma := 2$;
  $count := 0$;
  while x is not a feasible flow **and** $z < UB$ **and** $count < T_{max}$ **do**
    $\theta := \sigma[UB - L(\lambda)]/||\delta||^2$;
    $\lambda := \lambda + \theta\delta$;
    $c_{ij} := \lambda_i \alpha_{ij} + \lambda_j \gamma_{ij} \qquad \forall (i,j) \in A$;
    $x :=$ the optimal solution of the minimum cost flow problem in $G(x)$;
    $z := -x_{ts} + \sum_{(i,j)\in A} c_{ij}$;
    $count := count + 1$;
    if z has not changed in T_{lim} iterations **then**
      $\sigma := \sigma/2$;
    end if
  end while
  return (z, x);
end

Figure 2: The procedure *solve_lagrangian_dual* of the branch and bound algorithm

The pseudocode for the subgradient method to solve the Lagrangian dual problem is given in Figure 2. The procedure *solve_lagrangian_dual* takes an initial flow x and its corresponding residual network $G(x)$, and returns $L^*(\lambda)$ and x that corresponds to the maximized Lagrangian dual. Note that the iterations of the subgradient method may be stopped whenever $L(\lambda) \geq UB$ as that means we cannot obtain a better feasible flow than the best existing flow from this branch and bound node. We describe the branch and bound algorithm in the next section.

THE BRANCH AND BOUND ALGORITHM

At the beginning of the branch and bound algorithm, we set arc capacities as described in the Lagrangian Relaxation Section and solve problem P2 with $\lambda = 0$. If the resulting solution is feasible, than it is an optimal solution to problem P1. If it is not feasible, we execute the subgradient procedure to solve the Lagrangian dual. If the resulting solution is feasible, we might have obtained the optimal solution. In order to determine if that is the case, we check the optimality conditions. Optimality conditions consist of primal feasibility and complementary slackness. Complementary

slackness means that the corresponding lagrange multiplier is zero if a capacity constraint is not binding; and the corresponding constraint is binding for any lagrange multiplier that is nonzero.

Lemma 2 (Optimality Conditions). *An optimal solution (x^*, λ^*) to the Lagrangian subproblem $L(\lambda)$ (problem P2) is also an optimal solution to the generalized node-constrained maximum flow problem (problem P1) if and only if:*

(i) $\alpha x^* + \gamma x^* \leq W$
(ii) $\lambda^*(\alpha x^* + \gamma x^* - W) = 0$ (Complementary slackness)

Proof: Condition (i) implies that x^* is a feasible flow. Since $L(\lambda)$ is a lower bound for the optimal flow value of problem P1, we have: $-x_{ts}^* \geq L(\lambda^*) = -x_{ts}^* + \lambda(\alpha x^* + \gamma x^* - W)$. By condition (ii), we have $-x_{ts}^* = L(\lambda^*)$. □

If the solution is feasible after the subgradient procedure, we keep it as a candidate for optimal solution and set its flow value as the upper bound (UB). We apply branching to the solution of the relaxed problem before the subgradient procedure (which was not feasible). If the solution is infeasible, we apply branching to this solution.

We branch as follows. We pick node $n \in N$ with highest capacity violation. We then find arc $(a, b) \in A$ with the highest workload on the node based on current flow. Then, we calculate the maximum amount of flow u_{ab}^x on this arc (based on current flows on other arcs of n). We create two branches: (1) we set the upper bound u_{ab} to u_{ab}^x; (2) we set a lower bound l_{ab} for the arc that is equal to $u_{ab}^x + 1$. Clearly, these two branches are mutually exclusive and divide the search region in two. Adding the lower bound in branch (2) will cause a slight change to the solution procedure for the minimum cost network flow problem. At the beginning, we will push flows on shortest paths from s to a and from b to t until satisfying the lower bound of the arc. We will not add the reverse arc (b, a) to the residual network and keep pushing flows from s to t until there are no augmenting paths in $G(x)$.

procedure *make_branches* (x, G);
begin
  $n := argmax_i\{\delta_i, \forall i \in N\}$;
  $(a, b) := argmax_{(i,j)}\{\alpha_{ij} * x_{ij} | i = n, x_{ij} > 0;\ \ \gamma_{ij} * x_{ij} | j = n, x_{ij} > 0\}$;
  $u_{ab}^x = \max\{0, x_{ab} - \lceil \delta_i/\alpha_{ab} \rceil \ \ if \ \ a = n;\ \ x_{ab} - \lceil \delta_i/\gamma_{ab} \rceil \ \ if \ \ \ \ b = n\}$;
  G_1 := G;
  G_2 := G;
  { *Create branch 1*}
    $u_{ab} = u_{ab}^x$ in G_1;
  { *Create branch 2*}
    $l_{ab} = u_{ab}^x + 1$ in G_2;
  $\mathcal{B} := \mathcal{B} \cup \{G_1\}$;
  $\mathcal{B} := \mathcal{B} \cup \{G_2\}$;
end

Figure 3: The procedure *make_branches* of the branch and bound algorithm

We then select an unexamined branch and bound node and repeat the same procedure until all of the created branches are fathomed, at which time we will have found the optimal solution to problem P1. If a branch has a lower bound greater than the best UB, that branch is fathomed since

algorithm *branch_and_bound*;
begin
  $UB := 0$;
  $\mathcal{B} = \emptyset$
  x := the optimal flow of problem P2 on G with $\lambda = 0$;
  if $\delta_i \leq 0$ for all $i \in N$ **then**
    Stop. x is optimal and $z^* := -x_{ts}$;
  else
    $x_0 := x$;
    $\{LB,\ x\}$:= *solve_lagrangian_dual* (G);
    if $\delta_i \leq 0$ for all $i \in N$ **then**
      if $\lambda(\alpha x + \gamma x - W) = 0$**then**
        Stop. x is optimal and $z^* := -x_{ts}$;
      else
        $UB := \min\{UB, -x_{ts}\}$;
        $x := x_0$;
        make_branches (x, G);
      end if
    end if
    while $\mathcal{B} \neq \emptyset$ $\hat{G} :=$ a branch and bound node from $\mathcal{B}$ **do**
      x := the optimal flow of problem P2 on $\hat{G}$ with $\lambda = 0$;
      if $\delta_i \leq 0$ for all $i \in N$ **then**
        Stop.
        $UB := \min\{UB, -x_{ts}\}$;
      else
        $x_0 := x$;
        $\{LB,\ x\}$:= *solve_lagrangian_dual* $(\hat{G})$
        if $LB < UB$ **then**
          if $\lambda(\alpha x + \gamma x - W) = 0$ **then**
            $UB := \min\{UB, -x_{ts}\}$;
            $x := x_0$;
          end if
          make_branches (x, G);
          $\mathcal{B} = \mathcal{B} \backslash \hat{G}$
        end if
      end if
    end while
  end if
end

Figure 4: The algorithm *branch_and_bound*

it cannot yield a better solution. If a branch yields a feasible solution and satisfies Lemma 2, then that branch is fathomed as well, as we can't find a better solution by further branching from that node. We update the best UB if this feasible solution has a larger flow value. The pseudocode for the branch and bound algorithm is shown in Figures 3-4.

CONCLUSION

In this research we propose a Lagrangian relaxation based branch and bound procedure to solve the generalized node-constrained maximum flow problem. At each node of the branch and bound tree, subgradient optimization method is applied to solve the Lagrangian dual problem to obtain a lower bound. The proposed method is especially attractive as the entire solution process reduces to a series of shortest path problems on progressively smaller networks, which can be solved efficiently. It is also attractive due to the fact that Lagrangian relaxation lower bounds are typically tighter than LP relaxation lower bounds. We describe the details of the algorithm in this paper. Future research will focus on computational testing of the proposed solution procedure. Since there is no special method for this problem in the literature, we will compare our algorithm with general integer programming solvers such as CPLEX.

REFERENCES

[1] Ahuja, R.K. and Orlin, J.B. A capacity scaling algorithm for the constrained maximum flow problem. *Networks*, 25:89-98, 1995.

[2] Ahuja, R.K. and Magnanti, T.L. and Orlin, J.B. *Network flows: Theory, algorithms and applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[3] Çalışkan, C. A Double Scaling Algorithm for the Constrained Maximum Flow Problem. *Computers and Operations Research*, 35(4):1138-1150, 2008.

[4] Çalışkan, C. On a Capacity Scaling Algorithm for the Constrained Maximum Flow Problem. *Networks*, 53(3):229-230, 2009.

[5] Çalışkan, C. A Specialized Network Simplex Algorithm for the Constrained Maximum Flow Problem. *European Journal of Operational Research*, 210(2):137-147, 2011.

[6] Çalışkan, C. A Faster Polynomial Algorithm for the Constrained Maximum Flow Problem. *Computers and Operations Research*, 39(11):2634-2641, 2012.

[7] Chen, S. and Saigal, R. A primal algorithm for solving a capacitated network flow problem with additional constraints. *Networks*, 7:59-79, 1977.

[8] Ford, L.R. and Fulkerson, D.R. Maximum Flow through a network. *Canadian Journal of Mathematics*, 8:399-404, 1956.

[9] Fulkerson, D.R. Increasing the capacity of a network: The parametric budget problem. *Management Science*, 5(1):473-483, 1959.

[10] Graves, G.W. and McBride, R.D. The factorization approach to large-scale linear programming. *Mathematical Programming*, 10:91-110, 1976.

[11] Hall, R.W. and Çalışkan, C. Design and Evaluation of an Automated Highway System with Optimized Lane Assignment. *Transportation Research Part C*, 7(1):1-15, 1999.

[12] Hall, R.W. and Lotspeich, D. Optimized lane assignment on an automated highway. *Transportation Research Part C*, 4(4):211-229, 1996.

[13] Klingman, D. and Russell, R. Solving constrained transportation problems. *Operations Research*, 23(1):91-106, 1975.

[14] Malek-Zavarei, M. and Frisch, I.T. A constrained maximum flow problem. *International Journal of Control*, 14(3):549-560, 1971.

[15] Mamer, J.W. and McBride, R.D. A decomposition-based pricing procedure for large-scale linear programs: an application to the linear multicommodity flow problem. *Management Science*, 46(5):693-709, 2000.